Software Testing (UE18CS400SB)
Unit 4

Aronya Baksy

November 2021

1 Acceptance Testing

- Testing done in accordance with customer specified criteria (mutually agreed)
- Done in the customer environment by customer-specified individuals
- Types of Acceptance testing:
  - User Acceptance Testing
  - Business Acceptance Testing
  - Contract Acceptance Testing
  - Regulations Acceptance Testing
  - Operational Acceptance Testing
  - α and β testing
- Approaches to acceptance testing:
  - **Design / Architecture based approach**: Full functionality, Cases that fail here may trigger review
  - **Business vertical / customized instance**: Typically for products / frameworks Customized instances are put through testing based on business domain
  - **Deployment focused**: Large applications tested in complex hw or sw environments
- Criteria for acceptance testing:
  - SPecifying the business requirements (may be functional or non functional)
  - May be process requirements (test coverage, training etc.)
- Acceptance test cases cover: Critical functionality, End-to-end scenarios. New functionalities – during upgrade, Legal / statutory needs, Functionality to work on a defined corpus
- Acceptance test execution: on site, in the presence of dev engineers, with proper documentation of the execution process
- Challenges: criteria not easy to specify, multiple iterations with multiple cust. representatives needed

2 Non-Functional Testing

- Testing the non-functional attributes (performance, reliability, scalability, load, security, compatibility and others) of the system
- Reasons:
  - Identify and correct design faults
  - Identify limits of the system
  - Whether the product can behave gracefully during stress and load conditions
  - To ensure that the product can work without degrading for a long duration
  - To compare with other products and previous versions
  - To avoid un-intended side effects.
- When the product has no basic issues and meets the minimum entry criteria, then the non-functional test can start.
- The non-functional testing is stopped when there is enough data to make a judgement. This must be aligned with release schedule

2.1 Types of Non-Functional Testing

2.1.1 Scalability Test

- Ability of a system to handle increasing amounts of work without unacceptable level of performance (degradation)
- Scalability may be vertical or horizontal
- The test cases will focus on to test the maximum limits of the features, utilities and performing some basic operations

2.1.2 Performance Test

- Evaluating response time of product to perform required actions under stated conditions
- comparison with different versions of same product and competitive products.
- Test cases focus on getting response time and throughput for different operations, under defined environment and load

2.1.3 Reliability Test

- Evaluate the ability of the product to perform it's required functions under stated conditions for a specified period of time or number of iterations
- The test cases will focus on the product failures when the operations are executed continuously for a given duration or iterations
- Focus on frequently used operations

2.1.4 Stress Test

- Evaluate a system beyond the limits of the specified requirements or environment resources to ensure the product behavior is acceptable.
- Well-designed system shows graceful degradation and safe behaviour
- System should show performance decrease when resource/load ratio reduces, and symmetric increases when load is reduced

2.1.5 Security Test

- Both static and dynamic in nature
- Test tools identify vulnerabilities at application level (access control, SQL injection, buffer overflow, encryption, API design)

2.1.6 Regression Test

- A **black-box** technique, that ensures that code change does not impact existing functionality
- Types of regression testing:
  - **Corrective**: No changes to requirements, reuse existing test cases
  - **Retest-all**: Re-doing all tests again, not advisable for minor changes
  - **Selective**: Using a subset of the test cases, analyze impact of new code on existing code
  - **Progressive**: When change in spec, new test cases, ensures that new features do not break existing ones
  - **Complete**: Used for major changes in code.
  - **Partial**: Tests issues when new code is added to already existing code, ensures that a system continues to work after adding new code
  - **Unit**: focus on a single unit, block all dependencies

Software Testing (UE18CS400SB)
Unit 5

Aronya Baksy

November 2021

1 Testing Tools

1.1 JUnit

- An open-source testing framework for Java projects, based off XUnit
- Promotes test-driven development
- The testing code is embedded into the main Java program and by using the assert statements, and the programmers knowledge on the expectant result, we can test each component of the project
- Annotations identify test methods. Assert statements identify the results
- Tests run automatically, instant feedback, progress reports for test suites
- JUNit terminology:
  - A **test case** tests a single method
  - A **unit test** tests all methods in a class
  - **Test suite** combines unit tests
  - **Test fixture** provide support for test cases (envt setup, teardown)
  - **Test runner** runs the test suite/test cases
  - **Test result** summarises the test info of the test suite
- Disadvantages of JUnit: can't do dependency testing, not suitable for high level testing, can't test multiple JVMs at once

1.2 JMeter

- An open-source GUI application (Apache project) used for load/performance testing of web applications (now supports many types of apps)
- Useful in testing the performance of both static and dynamic resources like files, Servlets, Perl scripts, Java Objects, Data Bases, Queries, FTP Servers and more
- Allows for distributed testing (using multithreaded framework) and supports plugin addition or removal as per requirements.
- Advantages: open source, GUI tool, load/stress/distributed test, robust reporting, multiple protocol support
- Disadvantages: Limited to web app testing, no JS support, high memory usage, no complex scenarios with JMeter thread group

1.3 MonkeyTalk

- Real and functional interactive tests, smoke tests for Android and iOS apps (available on Linux, MacOS and Win)
- Used to be free and open source before Oracle acquisition
- Generates result in HTML and XML formats, supports JUnit reporting
- MonkeyTalk works with Emulators, Simulators, Tethered and Actual Hardware devices

- Provides record and playback, can be used to validate controls/images/text or any property of an object, provides complete gesture support.
- Supports linked-in libraries and subprojects
- Components of MonkeyTalk:
  - MonkeyTalk IDE: Eclipse-based IDE, used to record/playback/modify and manage test suites
  - MonkeyTalk Agent: A platform specific library injected into the app for the tool to be able to recognize it and test it
  - MonkeyTalk Scripts: 3 types (simple, parameterized, data-driven), can be in either JS/MonkeyTalk/Tabular scripting languages
  - Advantages: easy to learn, cross platform, supports keyword-driven and data-driven concepts, JUnit reporting, testing of desktop and mobile apps
  - Disadvantages: Need to install agent, no support for HTML5 and embedded webpages, hard to test games, bug discovery may take time due to lack of predefined test

1.4 Appium

- An open source, cross-platform tool for automating native, mobile web and hybrid apps on Android, iOS and Windows desktop
- Appium philosophy:
  - Shouldn't have to recompile or modify app to test it
  - Shouldn't be locked into a specific lang or framework for testing
  - Don't reinvent the wheel in terms of API design
  - Open source in name and spirit
- Appium uses client-server architecture. Appium server exposes REST API, that accepts request to start a test session and responds with a session ID
- Desired capabilites are sent in the request (as a JSON object) to identify the parameters for the test (e.g.: test platform iOS, allow Safari popups, etc.)
- Appium server runs the test server. It is written in node.js and available as an NPM package. Appium desktop is a GUI wrapper for the Appium server
- Appium client libraries in multiple languages support the Appium extended WebDriver protocol.
- Advantage: language agnostic, simple to use, same test base for multiple platforms
- Disadvantage: Slower than Espresso/XCUITests, complex to test cross-platform apps

1.5 Robotium

- Test framework for native and hybrid Android apps
- Test cases for automating GUI, functional, system, acceptance tests that can run on real device or emulator
- Based on JUnit and can be integrated with build tools like Maven and ANT
- Advantage: easy black-box testing, readable test cases, integrate with Maven, Ant, Gradle and other CI tools, works with APK and source code, IDE plgin available
- Disadvantage: one device, one app, one process at a time, not cross-platform (only for Android)

1.6 Selenium

- A portable testing framework that supports record-playback tools
- 4 components: Selenium IDE, Selenium remote control, WebDriver, Selenium Grid
- Selenium IDE: A GUI extension for Firefox/Chrome browsers that allows for recording, editing and debugging of functional tests
- Selenium WebDriver: used for automating web-based application testing to verify that it performs expectedly, run using test scripts

- Selenium Grid: tool used together with Selenium RC to run parallel tests across different machines and different browsers all at the same time
- Advantage: FOSS, multi browser and multi platform support, extensible, run tests in parallel
- Disadvantage: Only for web apps, no built-in object repository, slower, less support for data-drivent testing

1.7 Selendroid

- Selendroid is a test automation framework which drives off the UI of Android Native and hybrid applications (apps) and the mobile web.
- Android APK file must exist on the machine, where the selendroid-standalone server will be started to allow a customized selendroid-server for the app under test (AUT)
- Main selling point over Appium is **backward compatibility** (Selendroid supports Android API versions 10 onwards)
- Advantage: gesture support, multi-device support, hot plugging, inspector simplifies UI test development
- Disadvantage: need to recompile app for testing, complex for mobile webapps

1.8 Magneto

- Magneto is an open source test automation framework that allows to write smart and powerful tests for Android apps.
- Magneto is written in Python for Android devices.
- It utilizes the uiautomator tool via a Python wrapper and pytest as a test framework.
- Magneto can be triggered from CLI, IDE and CI tools

2 Defect Management

- Need a place to track and store all the defects logged during various testing phases and cycles
- Consider basic features like ticket statuses, email notifications, and the overall usability/experience
- Scalable, customizable, work with source control tool, Reporting capabilities